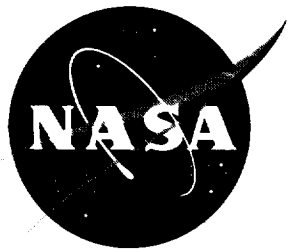

NASA Technical Paper 3551

CORSSTOL: Cylinder Optimization of Rings, Skin, and Stringers With Tolerance Sensitivity

J. Finckenor and M. Bevill
Marshall Space Flight Center • MSFC, Alabama

National Aeronautics and Space Administration
Marshall Space Flight Center • MSFC, Alabama 35812

May 1995

TABLE OF CONTENTS

	Page
INTRODUCTION	1
OUTPUT INTERPRETATION	1
PROGRAM LOGIC	6
CORSSTOL APPLIED	9
CONCLUSION	11
REFERENCES	12
APPENDIX A – Sample Input	13
APPENDIX B – Sample Output File	15
APPENDIX C – C Language Source Code	19

PRECEDING PAGE BLANK, NOT FILMED

PAGE ii INTENTIONALLY BLANK

LIST OF ILLUSTRATIONS

Figure	Title	Page
1.	Stringer geometry	3
2.	Main program routine	7
3.	Evaluate with tolerances	8
4.	Output tolerance calculations	10

LIST OF TABLES

Table	Title	Page
1.	Problem definition	2
2.	Initial optimized solution	4
3.	Toleranced optimization solution	5
4.	GSLOPE array	5
5.	Additional tolerance	6
6.	New design variable ranges	7
7.	Effect of tolerance variation	11

PRECEDING PAGE BLANK-NOT FILMED

PAGE INTENTIONALLY BLANK

TECHNICAL PAPER

CORSSTOL: Cylinder Optimization of Rings, Skin, and Stringers With Tolerance Sensitivity

INTRODUCTION

CORSSTOL builds on the analysis and numerical optimization routines used in CORSS,[1] however, CORSS is not required to understand the method that was used to study the tolerance sensitivities. CORSS is a skin-stringer analysis program tied to a commercially available numerical optimizer, "Design Optimization Tool" (DOT).[2] Either hat or I-stringers can be optimized. CORSSTOL is capable of working with any set of consistent units (although output currently better accommodates English units).

Designs provided by CORSS yield precise solutions with constraints close to 0, i.e., they are very close to being violated. This means that the final design leaves little or no room for error. For practical purposes, an optimum design should be robust enough to handle a range of dimensions while keeping the weight of the structure at a value not far from the absolute optimum (i.e., keep the weight close to the weight CORSS calculated using the optimum dimensions with no tolerances). CORSSTOL uses this concept to determine an optimum weight and a worst case weight given a set of tolerances for the design variables. CORSSTOL assumes that the design variables are the minimum material condition and that the design variables with the user input tolerance bands are the maximum material condition.

The problem statement for the untoleranced optimization is:

$$\begin{aligned} \text{Minimize} &: f(X_i) = \text{weight} \\ \text{Subject to} &: G_j(X_i) \le 0 \\ \text{and} &: XL_i \le X_i \le XU_i \ . \end{aligned}$$

The problem statement for the toleranced optimization is:

$$\begin{aligned} \text{Minimize} &: f(X_i+Tol_i) = \text{weight} \\ \text{Subject to} &: GW_j(XW_{i,j}) \le 0 \\ \text{and} &: XL_i \le X_i \le XU_i \ , \end{aligned}$$

where (X_i+Tol_i) is the maximum material condition. $XW_{i,j}$ is the worst case set of design variables and tolerances for each constraint. The individual terms of XW are either X or $X+Tol$, depending on whether an increase in the design variable increases (makes less conservative) or decreases (makes more conservative) the given constraint, respectively.

OUTPUT INTERPRETATION

This section deals with the output provided by the CORSSTOL program. The initial output is a repeat of the user provided input values (table 1). The sample case is a 48-in long cylinder with a 165.5-in radius. The material is aluminum 2219-T87. There are no intermediate rings. The stiffeners are I-stringers (fig. 1), and elastic buckling of the stringer webs is allowed. Elastic skin buckling is

allowed above the given load, as long as all other constraints are satisfied with a 1.4 safety factor when the skin is buckled. The number of stringers has been set to exactly 300, and the other design variable are optimized around that.

Table 1. Problem definition.

CORSS - Cylinder Optimization of Rings, Skin and Stringers
NASA/MSFC/ED52 - Structural Development Branch
Jeff Finckenor, Sep. 1993, ver. 2.1
Copyright (c) 1994 National Aeronautics and
Space Administration. No copyright claimed in
USA under Title 17, U.S. Code. All other rights
reserved.

CORSSTOL Sample Case

********** INPUT VALUES **********

FLAGS
- IPRINT = 0 — No screen output by DOT
- METHOD = 0 — Modified method of feasible directions
- SSP = 0 — Final CORSS output only

Material Properties
- nu = 0.330 — Poisson's ratio
- E = 1.080E+007 — Young's modulus
- SM = 4.000E+006 — Shear modulus
- rho = 0.111 — Density
- Stu = 63000.0 — Ultimate tensile stress
- Scy = 53000.0 — Yield compressive stress

Cylinder Geometry
- r = 165.50 — Radius
- l = 48.00 — Length
- fwt = 0.00 — Additional weight

Stringer Parameters
- stype = I — I stringers

Local elastic buckling IS ALLOWED
increase m from 1 while coupled buckling stress decreases
- alp = 0.000 — Web angle
- ll = 0.000 — Bottom flange thickness
- MZs = 1 — Stringers external

Ring Parameters
No Rings

Loads
- F = 410481.9 — Axial compression
- M = 9.152E+007 — Bending moment
- V = 98372.0 — Shear force
- Pa = 0.000 — Axial pressure component
- Ph = -0.087 — Hoop pressure component
- sf = 1.40 — Safety factor
- sfp = 1.00 — Plate buckling safety factor

Design Variables

Var.	Minimum	Initial	Maximum	Name
h	0.5000	1.5000	8.3750	Stringer height
l2	0.0400	0.0500	0.4075	Top flange thickness
tst	0.0400	0.0500	0.4075	Web thickness
Nst	300.0	300.0	300.0	Number of stringers
t	0.0400	0.0500	0.4075	Skin thickness
W	0.5000	2.5000	8.3750	Stringer width

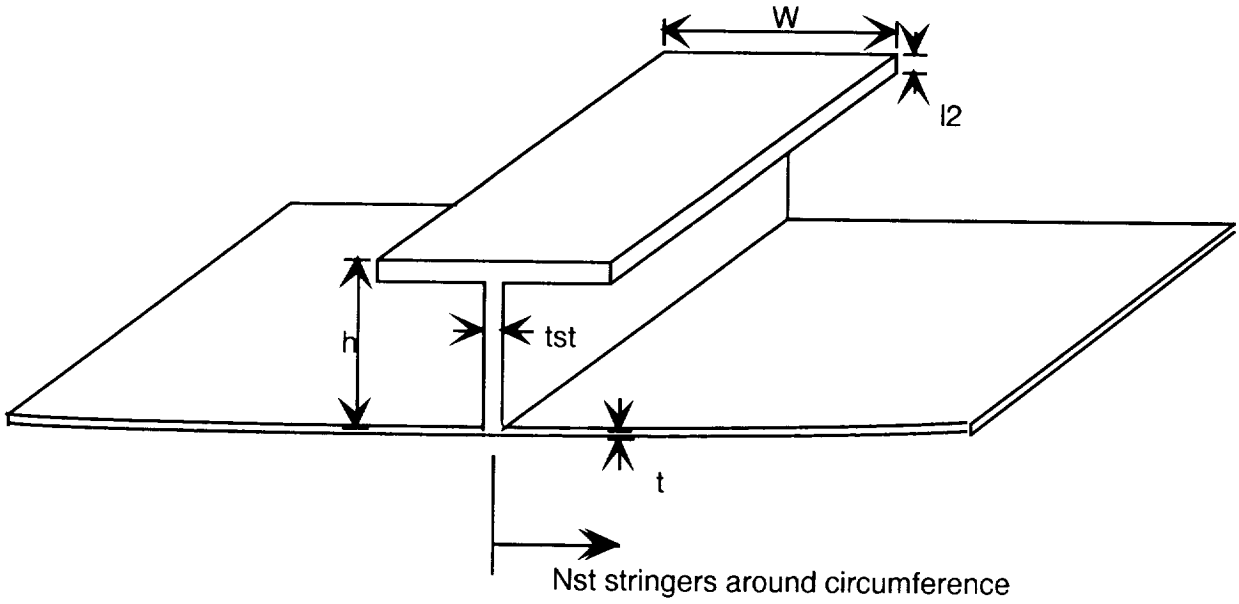

Figure 1. Stringer geometry.

CORSSTOL iterates through calls to DOT and the analysis until an optimized solution, without tolerances, is reached (table 2). This design case took four DOT iterations to optimize. Each DOT iteration involves multiple analysis calls to numerically calculate the function gradients and jump to a new design point. The optimized weight, without tolerance considerations, is given, along with the final design variable values. The stringer properties and required end ring properties are provided. The design constraints with their values are given. The convention is for a satisfied constraint to be negative.

The skin buckling constraint is a sum of stress ratios that must be less than 1. In this case, skin buckling is a driving constraint since the constraint value, G, is only slightly negative. The value for G is the sum of ratios minus 1. Coupled buckling stress is a failure mode in which the top flange of the I-stringer rolls in relation to the skin. This and the remaining constraints are calculated by: (critical stress)/(allowable stress) – 1. Stringer crippling stress is a material failure at the corners of the stringer due to the uneven stress distribution along the stringer cross section. Column stress is the Euler column buckling value of the stringer, taking into account any helpful effective skin. The stress constraint compares the material allowable with the distributed stress on the cylinder as a function of axial force and bending moment. Critical column buckling and general cylinder buckling are related. The general cylinder buckling analysis smears the stringers into an equivalent orthotropic shell and calculates the critical shell buckling load. If the skin buckles, this is not valid and is not used. If the skin does not buckle, then the more favorable of the stringer column buckling and the general cylinder buckling is used. This reflects the ability of the stringers to carry load without skin in some cases, and for the skin to transfer load between stringers in other cases.

After the initial optimization of the cylinder, the toleranced optimization begins. The toleranced optimization routine could be used from the initial values, but it is best to start from a near-optimum solution to reduce the program run time. During the initial optimization, it takes one skin stringer analysis to provide constraint values for a given set of design variables. For the tolerance optimization, it can take up to seven analyses for one set of constraints. An initial optimum analysis is required, plus an analysis for the perturbation of each of six design variables.

Table 2. Initial optimized solution.

```
Number of Iterations to Optimize = 4
CORSSTOL Sample Case

---------------------------------------------------------------------

Cylinder Weight = 595.5
  (Skin: 353.8, Stringers: 241.7, Rings: 0.0, Flanges: 0.0)

h    = 1.4064,     Stringer height
12   = 0.0400,     I top flange thickness
tst  = 0.0400,     I stringer web thickness
Nst  = 300.0,      Number of stringers, b = 3.4662
t    = 0.0639,     Skin thickness
W    = 2.4138,     I stringer width

Stringer: I = 0.025774, J = 0.000081, Z = 1.164139, A = 0.151207
End ring I should be at least 10.8982
End ring area should be at least 0.000439295*(r+Z)
                                                        G[ ] value
Skin: (shear ratio)**2 + Stress ratio      = 0.99470 < 1   -0.00530

Applied column stress (SF = 1.40)          = 19,343.3
   Critical coupled buckling stress (m = 1)= 19,447.4      -0.00535
   Critical stringer crippling stress      = 63,000.0      -0.69296
   Critical column stress                  = 19,450.6      -0.00552

Applied Von Mises stress (SF = 1.40)       = 19,187.4
   Yield compressive stress                = 53000.0       -0.63797

General cylinder buckling controlled by critical column buckling
```

The toleranced solution output (table 3) starts with a repetition of the user input tolerance ranges. Then output is the weight of the cylinder at the minimum material condition (optimum), the maximum material condition (maximum), and the difference (delta).

The design variables are listed with the optimum value, the maximum value (OPT+TOL column), the weight sensitivity to that variable(*dWt/dDV*), and the weight penalty (*dWt*) of the given tolerance. The maximum value is only the optimum value with the input tolerance added. The weight penalty is the weight of a cylinder with that design variable at its maximum and all the others at their minimum. The weight sensitivity is *dWt* divided by the input tolerance. For the sample case all tolerances were set at 0.01 in. In instances where a tolerance is given as 0, a 1-percent variation is used for the *dWt/dDV* calculation, and elsewhere in the program where a variation is needed.

All the optimization design values are length dimensions except the number of stringers. To be consistent with the length dimensions, the stringer spacing, *b*, was toleranced in the analysis instead of the number of stringers, Nst. Stringer spacing is the distance along the skin between the same point on adjacent stringers. The equation $b = (2\pi r)/Nst$ is used extensively throughout the code to relate the variables. Perturbations in *b* cause floating point variations in the number of stringers. However, DOT is a gradient optimizer, and *Nst* is handled as a floating point instead of an integer.

The GSLOPE array (table 4) provides the sensitivities of each constraint to each design variable. The minimum design variables are analyzed to create an *INITG* vector. Then each design variable is sequentially perturbed by its tolerance to give a new *G* vector. GSLOPE is equal to the difference between *INITG* and *G*, divided by the tolerance.

Table 3. Toleranced optimization solution.

---Tolerance Sensitivity Output for CORSS---

Design Variable Tolerances

htol =	0.0100	Stringer height tolerance
l2tol =	0.0100	Top flange thickness tolerance
tsttol =	0.0100	Stringer web thickness tolerance
Nsttol =	0.0100	Stringer spacing tolerance
ttol =	0.0100	Skin thickness tolerance
Wtol =	0.0100	Top flange width tolerance

NEW OPTIMUM ANALYSIS

SENSITIVITY ANALYSIS RESULTS

	Optimum	Maximum	Delta
Cylinder Weight	594.70	711.99	117.30

Design Variable Solutions

DV	Optimum	OPT+TOL	dWt/dDV	dWt
h =	1.40639	1.41639	63.7512	0.6375
l2 =	0.04000	0.05000	3,783.3191	37.8332
tst =	0.04000	0.05000	2,177.7588	21.7776
(b =	3.47622	3.46622)		
Nst =	299.13699	300.00000	0.8056	0.6953
t =	0.06385	0.07385	5,538.1045	55.3810
W =	2.41377	2.42377	63.7512	0.6375

Number of Stringers, Nst = 2*pi*r/b, b = stringer spacing

Small changes in b change the theoretical number of stringers used in the analysis. As stringer spacing increases, the number of stringers decreases. The design will have an integer number of stringers, but the effect of the spacing tolerance is shown here. Any differences between the sum of the dWt values and deltaWt are due to the weight function not being linear. All calculations are based on taking differences about the optimum.

Table 4. GSLOPE array.

GSLOPE ARRAY

DV	dSkB/dDV	dShB/dDV	dStr/dDV	dCrp/dDV	dICB/dDV
h	-0.1104	-1.4281	-0.0463	-0.0390	0.0099
l2	-6.4130	-11.8218	-2.6923	-2.2653	-10.6503
tst	-3.6700	-1.4574	-1.5418	-1.2971	-2.4030
b	0.6387	0.1288	0.0507	0.0426	0.1378
t	-39.6311	-9.0021	-4.8850	-4.1480	-13.4095
W	-0.1080	-0.2092	-0.0453	-0.0381	-0.9595
G[i]	-0.00008*	-0.00046*	-0.63296	-0.68795	-0.00518*

* Denotes the driving constraints, which must be negative to be satisfied
GSLOPE = [G(1.01*DV)-G(DV)]/(1.01*DV-DV)
negative values of d( )/dDV indicate a safer design for an increase in DV
DV = design variable G[i] = constraint value
SkB = skin buckling ShB = shell buckling
Str = stress Crp = stringer crippling
ICB = 'I'-stringer coupled buckling

A negative value in GSLOPE indicates the design gets safer as the variable is increased. This is because constraints must be less than 0 to be satisfied.

The signs of the GSLOPE values are used to determine the worst-case condition for each individual constraint. Design variables with a negative GSLOPE are used at the given value, variables with a positive slope have their tolerance added. The I-stringer coupled buckling constraint (*dICB/dDV*) has negative GSLOPE values for 12, *tst*, *t*, and *W* and positive values for *h* and *b*. The constraint value used in the optimization is calculated using the given values of 12, *tst*, *t*, and *W* and values with the tolerances added for *h* and *b*.

The additional tolerance array (table 5) checks to see if there is a safe area outside the specified range. This information feeds directly into the NEW RANGES information that follows.

The DOT optimizer finds solutions with one or more constraints near, but not exactly zero. A linear approximation is used to find the value of the design variable at which the related constraint is exactly equal to zero. Negative numbers have no physical meaning, other than that the dimension under consideration has no significant effect on that constraint. Also, constraints that are not driving the design will tend to have negative additional tolerance values since they are starting at a relatively large negative number.

Table 5. Additional tolerance.

ADDITIONAL TOLERANCE ARRAY

DV	SkB	ShB	Str	Crp	ICB
h	1.4057	1.4061	–12.2652	–16.2549	1.9418
l2	0.0400	0.0400	–0.1951	–0.2637	0.0395
tst	0.0400	0.0397	–0.3705	–0.4904	0.0378
b	3.4764	3.4798	15.9694	19.6152	3.5138
t	0.0639	0.0638	–0.0657	–0.1020	0.0635
W	2.4130	2.4116	–11.5513	–15.6270	2.4084

The ADDTOL array contains the values of the design variables at which the constraint is equal to 0, assuming linear relationships.
ADDTOL = DV – G(DV)/GSLOPE
Negative values should be disregarded, since they indicate the design variable can be brought past zero without violating that particular constraint.

New ranges (table 6) use the additional tolerance array to see if there is any benefit that can be gained for the design. For the sample case the optimum calculated value of *h* is 1.40639. Because of numerical limitations on making the constraints exactly equal to zero, this can be reduced slightly to 1.40607. However, h could be allowed to increase to 1.94183 without any constraints being violated, and only costing a maximum of 33.5 lb. While over 0.5 in is an excessively large tolerance for a 1.5-in stringer, the designer is now aware that this tolerance can easily be made very loose. It is important to note that the new tolerance band is calculated with a linear approximation. If a wider tolerance is desired, it is important to rerun the optimization with the larger tolerance band as input. Since the constraints are calculated at the bounds of the given tolerance, the design will always have a positive margin of safety for the given tolerance range.

PROGRAM LOGIC

Note that all of the sensitivities calculated by CORSSTOL are based on individual design variables. There is no consideration of interaction among the design variables. This assumption should be acceptable as long as variations are small.

Additional code shown in appendix A but not included in the following flow diagrams is to avoid repetition of an analysis, to normalize the variables, or to prevent divide by 0 errors.

Table 6. New design variable ranges.

NEW RANGES for the Design Variables (from the Additional Tolerance array)

	Below Optimum ----			Above Maximum ----			
DV	Lower Limit	Weight Savings	Added Toler.	Upper Limit	Weight Penalty	Added Toler.	New Tol. Band
h	1.40607	0.0206	0.00032	1.94183	33.5	0.52544	0.53576
12	0.03999	0.0478	0.00001	No constraint limits the upper bound			
tst	0.03998	0.0481	0.00002	No constraint limits the upper bound			
Nst	299.12608	0.0088		No constraint limits the upper bound			
(b	No limit on the lower bound		3.47635			0.00013	3.47635)
t	0.06385	0.0113	0.00000	No constraint limits the upper bound			
W	2.41302	0.0479	0.00075	No constraint limits the upper bound			

The only penalty for increasing design variables with no upper limit is an increase in weight. Note that an unsatisfied constraint (G[i]>0) will cause unrealistic results. For a feasible solution, no constraints will be violated as long as the DVs are held within the limits specified in this chart when ASSUMING LINEAR RELATIONSHIPS.

Figure 2 shows the operation of the main routine of the program. After the parameters for DOT and the program are assigned, the initial evaluation is performed with the user input initial design variables, $X[i]$. The constraint array, $G[i]$, and the weight are calculated. The evaluation is looped with DOT which controls the numerical calculation of gradients and jumps to new points in the design space.

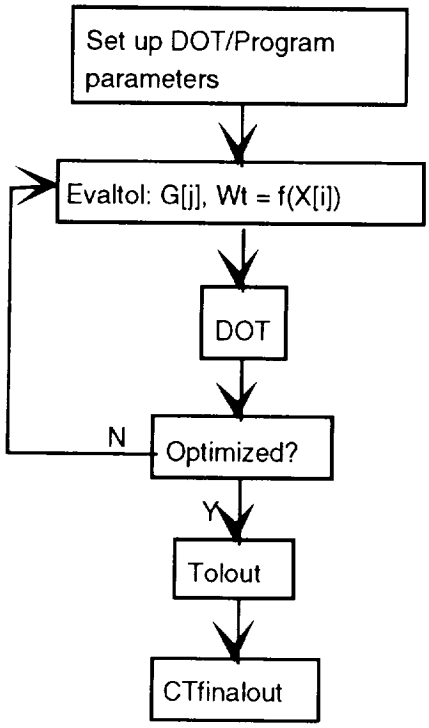

Figure 2. Main program routine.

DOT is a gradient optimizer that incorporates the gradients of the constraints so that it follows the constraint boundaries to an optimum point. A more thorough description of DOT can be found in references 1 and 3.

Once the optimum has been found and the loop exited, the sensitivities of interest to the user are calculated in "Tolout" and printed to an output file by "CTfinalout".

Figure 3 shows the method used in "Evaltol" to evaluate the constraints, $G[i]$ and the weight, OBJ. The design variables that DOT operates on are copied into vectors that will change during the evaluation. The *initX* array is used as a reference point, while *XW* is adjusted to provide a worst-case set of design variables. The initial constraint array, *initG*[*j*], is calculated using *initX*.

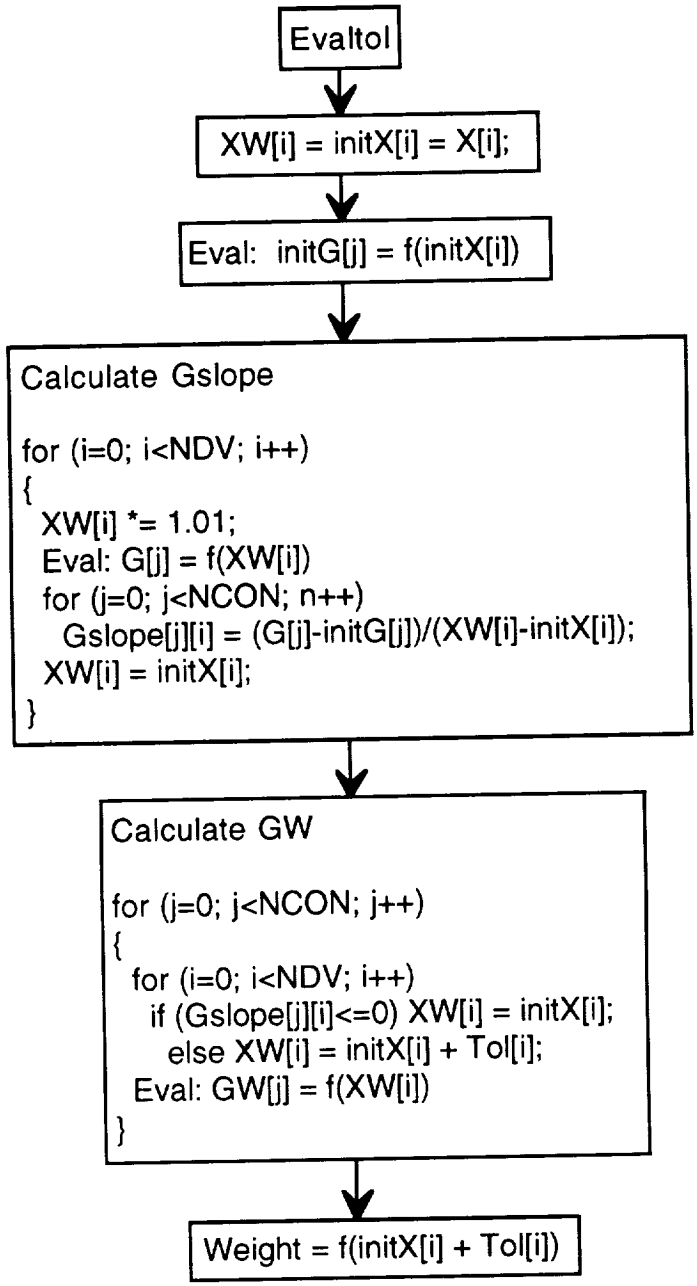

Figure 3. Evaluate with tolerances.

The GSLOPE array is calculated by sequentially perturbing each design variable and evaluating the constraints. The changes are referenced back to the initG and initX vectors to provide a linear approximation of the slopes at that point.

GW is the worst-case set of constraints and is the only constraint vector that DOT sees. In order for a constraint to be satisfied, it must be negative. If GSLOPE for a given constraint and design variable is negative that means increasing the variable makes the design more conservative. Positive values for GSLOPE indicate that the design is less conservative as the design variable increases. Based on the sign of GSLOPE, the *XW* vector is filled either with the minimum value allowed or the minimum plus the given tolerance (the maximum). Since increases in design variables may improve some constraints while decreasing others, each constraint value is evaluated for its own worst case set of design variables.

Finally the maximum material condition weight is calculated to act as the objective function.

Figure 4 shows how "Tolout" calculates the sensitivities of interest to the user. The DOT design variables, *X*, are copied into a working vector, *XW*. The optimum (minimum material condition) and maximum (maximum material condition) weights are calculated. Then a working vector of tolerances, *TW*, is set to 0.

The weight is calculated as the tolerances are sequentially set to their user input value. From that the weight difference due to each variable and a linear approximation of the slope of the weight are calculated.

The final constraint values are never exactly equal to zero, and noncritical constraints may be much less than zero. The AddTol array contains the design variable values at which the constraint is zero. Note that this assumes a linear variation, which will generally only be accurate if the constraint is near zero. Other values should be used as an indicator, but not considered a precise value.

New tolerance bounds are calculated using the AddTol array. The AddTol values for a given design variable are directly compared to find the minimum and maximum.

CORSSTOL APPLIED

The benefits of CORSSTOL can be seen in table 7. The optimum design variables for each set of tolerances are shown. The tolerances used were 0.030 applied to each variable, 0.010 to each variable, a varying tolerance, and a zero tolerance. The design variable values shown are the optimum (except *b*, which was fixed), to which the tolerances are added. The column with the varying tolerance shows the optimum variable plus the tolerance band for that variable.

If the tolerances of a given cylinder are assumed to be normally distributed over the whole surface, then the average of the minimum and maximum weights is the expected weight. The "Increase" row is the percent increase of the average cylinder weight over the zero tolerance weight.

The overall 0.030 tolerance increases the weight of the cylinder 29 percent above the optimum, which is unacceptable for a light weight structure. Reducing the overall tolerance to 0.010 reduces the weight increase to 9.6 percent. By using the information provided by the 0.030 and 0.010 runs, the tolerances can be chosen so the weight increase is only 4 percent.

The weight sensitivity of stringer spacing, *b*, is very low and it has a very small impact on weight. The stringer height, *h*, and width, *W*, have slightly higher weight sensitivities, but the overall effect on weight is still small. Since the effect on the weight from these dimensions is small, they can be given a fairly loose 0.060 (or ±0.030) tolerance.

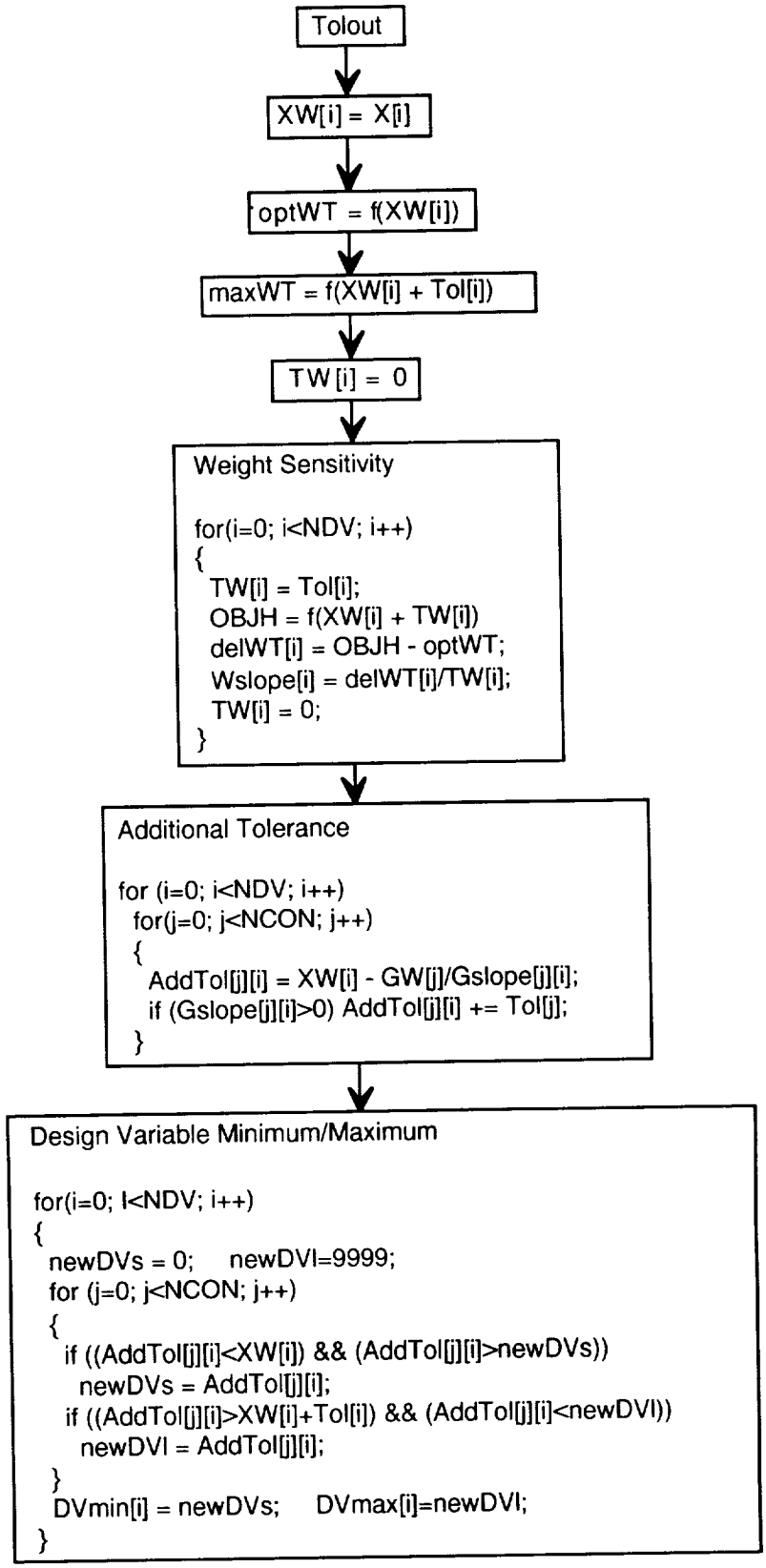

Figure 4. Output tolerance calculations.

The stringer thicknesses, l2 and *tst*, have a much greater weight sensitivity and impact on the overall weight. If the assumption is that the stringers are integrally milled out of a plate, it may not be possible to get very tight tolerances. However it is still necessary to keep them tight because of the weight. A compromise tolerance of 0.005 was chosen, although the actual value would be dependent on the capability of the available shop equipment.

Skin thickness, *t*, has the highest weight sensitivity and impact, so it is most important to keep its tolerance as tight as possible. With the assumption of an integrally milled plate, the skin thickness would be easier to control than the stringer thicknesses because the skin cannot get pushed away by the milling head. A tolerance of 0.002 (±0.001) was chosen. Again, this would be dependent on the capability of the equipment.

Table 7. Effect of tolerance variation.

	Tol[i]=.030	**Tol[i]=.010**	**Tol[i] varies**	**Tol[i]=0.000**
h	1.4078	1.4064	1.4120 + 0.060	1.4050
l2	0.0400	0.0400	0.0400 + 0.005	0.0400
tst	0.0400	0.0400	0.0400 + 0.005	0.0400
b	3.4662	3.4662	3.4662 + 0.060	3.4662
t	0.0642	0.0639	0.0647 + 0.002	0.0640
W	2.4093	2.4138	2.4130 + 0.060	2.4119
Min Weight	594.78	594.70	596.04	596.10
Max Weight	947.47	711.99	644.20	
Avg Weight	771.13	653.35	620.12	
Increase	29.39%	9.60%	4.03%	

Use of CORSSTOL in this case has produced a design for a stringer-stiffened cylinder. Tolerances have been intelligently assigned so that noncritical areas have a very large range, and only the most critical dimensions have tight tolerances. Manufacturing costs can be concentrated on the critical dimensions to provide a good part for less cost.

CONCLUSION

CORSSTOL demonstrates a method that allows users to determine the sensitivity of their designs to dimensional tolerances. While the method does not provide values for the tolerances of the design variables, it does show the effects of user input tolerances. This method allows for tolerances to be assigned in a more analytic and functional manner, and gives insight to the designer for deciding if a part made out of tolerance is still safe.

REFERENCES

1. Finckenor, J., Rogers, P., and Otte, N.: "CORSS: Cylinder Optimization of Rings, Skin, and Stringers." NASA Technical Paper 3457, January 1994.

2.. Vanderplaats, G., and Hansen, S.: "Design Optimization Tool." VMA Engineering, 1985–1988.

3. Vanderplaats, G.N.: "Numerical Optimization Techniques for Engineering Design: With Applications." McGraw-Hill Book Company, 1984.

APPENDIX A

Sample Input

```
CORSSTol Sample
0          Screen output by DOT 0=none 7=most,IPRINT
0          METHOD 0,2=Mod.Meth.of Feas.Dirs.,2=Seq.Lin.Prog
0          Output, 0=final,1=final calcs.+0,2=dv/con+0,3=dv/con+1
.33        Poisson's Ratio
10.8e6     Young's Modulus, compressive
4.E6       Shear Modulus
.111       density
63000      Ultimate Tensile Strength, enter as positive
53000      Compressive Yield Strength, enter as positive
165.5      radius of cylinder
48         length of cylinder
0.0        extra nonoptimized weight
I          stringer type, 'H' for hat stringers, 'I' for I-stringers
Y          Allow Local Elastic Buckling [Y/N]
0          I-str coupled buckling flag, 0=find bucket, #=search m=1 to #
0.         web angle in degrees, 0 is perpendicular to skin
0.         hat to skin length (2/hat), or height of I bottom flange
1          1 for external stringers, -1 for internal
1.50       ring cross sectional area
63.0       ring Moment Of Inertia
-1.0       ring centroid location
58.0       ring Polar Moment Of Inertia
0          number of intermediate rings
410481.9   Applied Axial Force, positive is compressive
91520000   Applied Bending Moment, enter as positive
98372      Applied Shear Force, enter as positive
0.00       Axial Pressure (i.e. tank ullage) +internal
-.087      Hoop Pressure (i.e. ullage+head) +internal
1.4        overall safety factor
1.0        skin buckling safety factor
.5         h min, stringer height
1.5        h init
8.375      h max
0.01       h Tolerance
.04        12 min, hat: top flange length, I: top flange thickness
.05        12 init
.4075      12 max
0.01       12 Tolerance
0.04       tst min, hat: stringer thickness, I: web thickness
0.05       tst init
0.4075     tst max
0.01       tst Tolerance
300        Nst min, number of stringers
300        Nst init
300        Nst max
0.01       Nst Tolerance
0.04       t min, skin thickness
0.05       t init
0.4075     t max
0.01       t Tolerance
0.5        W min, hat: top flange thickness, I: width
2.5        W init, hat: set min,init and max to 0 for
8.375      W max,       hats of constant tst
0.01       W Tolerance
```

APPENDIX B

Sample Output File

```
CORSS - Cylinder Optimization of Rings, Skin and Stringers
     NASA/MSFC/ED52 - Structural Development Branch
     Jeff Finckenor, Sep. 1993, ver. 2.1
     Copyright (c) 1994 National Aeronautics and
     Space Administration. No copyright claimed in
     USA under Title 17, U.S. Code. All other rights
     reserved.

CORSSTol Sample

************************ INPUT VALUES *****************
FLAGS
     IPRINT = 0      No screen output by DOT
     METHOD = 0      Modified Method of Feasible Directions
     SSP = 0         Final CORSS output only

Material Properties
     nu    = 0.330              Poisson's Ratio
     E     = 1.080E+007         Young's Modulus
     SM    = 4.000E+006         Shear Modulus
     rho   = 0.111              Density
     Stu   = 63000.0            Ultimate Tensile Stress
     Scy   = 53000.0            Yield Compressive Stress

Cylinder Geometry
     r    =  165.50             Radius
     l    =  48.00              Length
     fwt =  0.00                Additional Weight

Stringer Parameters
     stype = I                  I Stringers
     Local Elastic Buckling IS ALLOWED
     increase m from 1 while coupled buckling stress decreases
     alp    = 0.000             Web Angle
     ll     = 0.000             bottom flange thickness
     MZs    = 1                 Stringers external

Ring Parameters
     No Rings

Loads
     F    = 410481.9            Axial Compression
     M    = 9.152E+007          Bending Moment
     V    = 98372.0             Shear force
     Pa   = 0.000               Axial pressure component
     Ph   = -0.087              Hoop pressure component
     sf   = 1.40                Safety factor
     sfp  = 1.00                Plate buckling safety factor

Design Variables
Var.      Minimum       Initial       Maximum       Name
h          0.5000        1.5000        8.3750       Stringer height
12         0.0400        0.0500        0.4075       Top flange thickness
tst        0.0400        0.0500        0.4075       Web thickness
Nst         300.0         300.0         300.0       Number of stringers
t          0.0400        0.0500        0.4075       Skin thickness
W          0.5000        2.5000        8.3750       Stringer Width
```


PRECEDING PAGE BLANK NOT FILMED

PAGE 14 INTENTIONALLY BLANK

```
Number of Iterations to Optimize = 4
CORSS using SGA optimizer

--------------------------------------------------------------------------------

Cylinder Weight = 595.5
 (Skin: 353.8, Stringers: 241.7, Rings: 0.0, Flanges: 0.0)

h   = 1.4064,      Stringer Height
12  = 0.0400,      I top flange thickness
tst = 0.0400,      I stringer web thickness
Nst =  300.0,      Number of Stringers, b = 3.4662
t   = 0.0639,      Skin Thickness
W   = 2.4138,      I stringer width

Stringer: I = 0.025774, J = 0.000081, Z = 1.164139, A = 0.151207
End Ring I should be at least 10.8982
End Ring Area should be at least 0.000439295*(r+Z)

                                                    G[] value
Skin: (Shear ratio)**2 + Stress ratio             = 0.99470 < 1     -0.00530

Applied Column Stress (SF = 1.40)                 =     19343.3
    Critical Coupled Buckling Stress (m =    1)   =  19447.4         -0.00535
    Critical Stringer Crippling Stress            =  63000.0         -0.69296
    Critical Column Stress                        =  19450.6         -0.00552

Applied Von Mises Stress (SF = 1.40)              =     19187.4
    Yield Compressive Stress                      =  53000.0         -0.63797

General Cylinder Buckling controlled by Critical Column Buckling


---Tolerance Sensitivity Output for CORSS---

Design Variable Tolerances
     htol   =              0.0100 Stringer Height Tolerance
     12tol  =              0.0100 Top Flange Thickness Tolerance
     tsttol =              0.0100 Stringer Web Thickness Tolerance
     Nsttol =              0.0100 Stringer Spacing Tolerance
     ttol   =              0.0100 Skin Thickness Tolerance
     Wtol   =              0.0100 Top Flange Width Tolerance

NEW OPTIMUM ANALYSIS

SENSITIVITY ANALYSIS RESULTS

                    Optimum    Maximum      Delta
Cylinder Weight          594.70     711.99     117.30


Design Variable Solutions
     DV         Optimum       OPT+TOL       dWt/dDV          dWt
     h    =     1.40639       1.41639       63.7512        0.6375
     12   =     0.04000       0.05000     3783.3191       37.8332
     tst  =     0.04000       0.05000     2177.7588       21.7776
    (b    =     3.47622       3.46622)
     Nst  =   299.13699     300.00000        0.8056        0.6953
     t    =     0.06385       0.07385     5538.1045       55.3810
     W    =     2.41377       2.42377       63.7512        0.6375
```

Number of Stringers, Nst = 2*pi*r/b, b = stringer spacing

Small changes in b change the theoretical number of stringers used in the analysis. As stringer spacing increases, the number of stringers decreases. The design will have an integer number of stringers, but the effect of the spacing tolerance is shown here. Any differences between the sum of the dWt values and deltaWt are due to the weight function not being linear. All calculations are based on taking differences about the optimum.

GSLOPE ARRAY

DV	dSkB/dDV	dShB/dDV	dStr/dDV	dCrp/dDV	dICB/dDV
h	-0.1104	-1.4281	-0.0463	-0.0390	0.0099
12	-6.4130	-11.8218	-2.6923	-2.2653	-10.6503
tst	-3.6700	-1.4574	-1.5418	-1.2971	-2.4030
b	0.6387	0.1288	0.0507	0.0426	0.1378
t	-39.6311	-9.0021	-4.8850	-4.1480	-13.4095
W	-0.1080	-0.2092	-0.0453	-0.0381	-0.9595
G[i]	-0.00008*	-0.00046*	-0.63296	-0.68795	-0.00518*

* Denotes the Driving Constraints, which must be negative to be satisfied
GSLOPE = [G(1.01*DV)-G(DV)]/(1.01*DV-DV)
negative values of d( )/dDV indicate a safer design for an increase in DV
DV = Design Variable G[i] = Constraint Value
SkB = Skin Buckling ShB = Shell Buckling
Str = Stress Crp = Stringer Crippling
ICB = 'I'-stringer Coupled Buckling

ADDITIONAL TOLERANCE ARRAY

DV	SkB	ShB	Str	Crp	ICB
h	1.4057	1.4061	-12.2652	-16.2549	1.9418
12	0.0400	0.0400	-0.1951	-0.2637	0.0395
tst	0.0400	0.0397	-0.3705	-0.4904	0.0378
b	3.4764	3.4798	15.9694	19.6152	3.5138
t	0.0639	0.0638	-0.0657	-0.1020	0.0635
W	2.4130	2.4116	-11.5513	-15.6270	2.4084

The ADDTOL array contains the values of the design variables
at which the constraint is equal to 0, assuming linear relationships.
ADDTOL = DV - G(DV)/GSLOPE
Negative values should be disregarded, since they indicate the design variable can be brought past zero without violating that particular constraint.

NEW RANGES for the Design Variables (from the Additonal Tolerance array)

	Below Optimum ----			Above Maximum ----			
DV	Lower Limit	Weight Savings	Added Toler.	Upper Limit	Weight Penalty	Added Toler.	New Tol. Band
h	1.40607	0.0206	0.00032	1.94183	33.5	0.52544	0.53576
12	0.03999	0.0478	0.00001	No constraint limits the upper bound			
tst	0.03998	0.0481	0.00002	No constraint limits the upper bound			
Nst	299.12608	0.0088		No constraint limits the upper bound			
(b	No limit on the lower bound			3.47635		0.00013	3.47635)
t	0.06385	0.0113	0.00000	No constraint limits the upper bound			
W	2.41302	0.0479	0.00075	No constraint limits the upper bound			

The only penalty for increasing design variables with no upper limit is an

increase in weight. Note that an unsatisfied constraint (G[i]>0) will cause unrealistic results. For a feasible solution, no constraints will be violated as long as the DVs are held within the limits specified in this chart when ASSUMING LINEAR RELATIONSHIPS.

APPENDIX C

C Language Source Code

FROM Include File CORSS.H
Analysis Code refered to but not shown here is included in NASA TP-3457

```
/* C language include files for standard function definitions */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

/* define pi for use in the program */
#define pi  3.141592654

/* Variable Structures
Many often passed values are grouped into structures of related variables
stiffness - the smeared orthotropic cylinder properties used in General
        Cylinder Buckling Calculations
        Ex, Ey, Exy - smeared extensional stiffnesses
        Gxy - smeared shear stiffness
        Cx, Cy, Cxy, Kxy - smeared coupling constants
        Dx, Dy, Dxy - smeared bending stiffnesses
material - the material properties from the input file
        E - Young's Modulus
        nu - poisson's ratio
        rho - density
        Scy - compressive yield strength
        G - Shear Modulus
        Stu - tensile ultimate strength
load - the loading parameters from the input file
        F - axial force, positive is compressive
        M - bending moment
        Pa - axial pressure (i.e. ullage), internal is positive
        Ph - hoop pressure (i.e. ullage+head), internal is positive
        sf - safety factor to be applied to loads
        sfp - skin buckling safety factor, assumed 1.0<=sfp<=sf
        V - shear force
stringer - the variables that define the stringers
        A - cross sectional area
        I - Moment of Inertia about the neutral axis parallel to the skin
        J - The polar moment of inertia, Ix+Iy
        N - the number of stringers (optimized design variable)
        Z - neutral axis distance from the skin
        alp - web angle
        11 - hats: stringer to skin length, 2/hat; I: bottom flange thickness
        MZs - Z multiplier, 1 for external stringers, -1 for internal
        h - stringer height (optimized design variable)
        12 - hat: top flange width, I: top flange thickness (design variable)
        t - skin thickness (optimized design variable)
        W - hat: top flange thickness, I: stringer width (design variable)
        b - stringer spacing
        la - projected length of the hat stringer web on the skin
        stype - stringer type identifier, H for hats, I for I's
ring - ring parameters from the input file
        A - cross sectional area
        I - moment of inertia about the neutral axis parallel to the skin
        J - Polar Moment of Inertia, Ix+Iy
        N - number of intermediate rings (not end rings)
        Z - neutral axis distance from the skin, + for external
        d - ring spacing
cylinder - cylinder geometry definitions
        fwt - additional weight, not optimized
        l - length of cylinder
```

```
      r - radius of cylinder
      t - skin thickness (optimized design variable)
*************************************************************************/
struct stiffness{float Ex, Ey, Exy, Gxy, Dx, Dy, Dxy, Cx, Cy, Cxy, Kxy; };
struct material {float E, nu, rho, Scy, G, Stu; };
struct load     {float F, M, Pa, Ph, sf, sfp, V; };
struct stringer {float A, I, J, N, Z, alp, 11, MZs, h, 12, t, W, b, 1a;
                 char stype;};
struct ring     {float A, I, J, N, Z, d; };
struct cylinder {float fwt, l, r, t; };


/*************************************************************************/
/*************************    file CT.C    ******************************/
/*************************************************************************/

/* MAIN starts the program, initialized the variables and calls the
      optimization routine
called from: DOS command line
calls      : readinput, pinput, initDVs, CALLeval, and corsstol
returns    :
    argc, argv - the number and array of command line parameters,
    G - array of constraint values,
    hap - position of stringer height in X
    i - index,
    in, out - handles for the input and output files,
    IPRINT - DOT output flag,
    12ap - position of top flange length in X
    LEB - Y/N flag to allow local elastic buckling of the stringer segments
    METHOD - DOT method flag,
    mflag - flag to control search of I-coupled buckling
    NCON - number of constraints,
    NDV - number of design variables,
    Nstap - position of number of stringers in X
    OBJ - weight,
    SSP - CORSS output flag
    tap - position of skin thickness in X
    Title - title of run,
    Tol - array of tolerances associated with X
    tstap - position of stringer thickness in X
    Wap - position of I width: H top flange thickness in X
    X, XL, XU - array of design variables, and lower and upper limits,
    Xinit - array of initial variables for normalizing,
*************************************************************************/


/* comlineerr is called if there is an error with the command line. It
      prints the correct usage message and exits the program
called from: MAIN
calls      :
returns    :
*************************************************************************/
void comlineerr();



/* initDVs controls the removal of variables from the design variable arrays
called from: MAIN
calls      : Xadust
returns    : modified X,XL,XU,Xinit arrays, and hap,12ap,tstap,Nstap,tap,Wap
               indices and the new number of design variables
      i - incrementing index
      NDV - number of design variables
      pos - current X array position
*************************************************************************/
long int initDVs(float XL[], float X[], float XU[], int *hap, int *12ap,
      int *tstap, int *Nstap, int *tap, int *Wap, float Xinit[],
```

```
        long int ndv);


/* Xadjust manipulates the design variable arrays so that only the needed
        design variables are iterated upon
called from: initDVs
calls       :
returns     : next available design variable array position
        k - index
*************************************************************************/
int Xadjust(int newpos, int pos, long int *NDV, float X[], float Xinit[],
            float XL[], float XU[]);


/*************************************************************************/
/*************************    file CTOPT.C    ****************************/
/*************************************************************************/

/* DOT is the numerical optimizer program
called from: ctopt
returns     : new values for X
*************************************************************************/
void fortran DOT(long int *INFO,   long int *METHOD, long int *IPRINT,
                 long int *NDV,    long int *NCON,   float    X[],
                 float    XL[],    float    XU[],    float    *OBJ,
                 long int *MINMAX, float    G[],     float    RPRM[],
                 long int IPRM[],  float    WK[],    long int *NRWK,
                 long int IWK[],   long int *NRIWK);


/* CALLeval uses the X array and the 'ap' variable to prepare the variables
        for the analysis of EVAL
calls    : eval
returns :
*************************************************************************/
void CALLeval(struct stringer S, struct ring R, struct material M,
        struct load L, struct cylinder C, float G[], float X[],
        float Xinit[], float *OBJ, FILE *out, int SSP, int hap, int l2ap,
        int Nstap, int tap, int tstap, int Wap, int mflag, char LEB);



/* ctopt controls the optimization calls to eval and DOT
called from: MAIN
calls       : CALLeval and DOT
returns     :
        INFO - DOT completion flag
        IPRM, IWK, RPRM, WK - DOT working arrays
        MINMAX - DOT minimization/maximization flag
        NRIWK, NRWK - IWK and WK sizes
*************************************************************************/
void ctopt(int SSP, FILE *out, float X[], float Xinit[], int hap,
        int l2ap, int tstap, int Nstap, int tap, int Wap, struct stringer S,
        struct ring R, struct material M, struct load L, struct cylinder C,
        float G[], float *OBJ, long int *METHOD, long int *IPRINT,
        long int *NDV, long int *NCON, float XL[], float XU[], int mflag,
        char LEB);


/*************************************************************************/
/*************************    file EVAL.C     ****************************/
/*************************************************************************/

/* EVAL is the function that controls the analysis
called from: CALLeval
calls       : stringer, skinbuck, getScrip, colstress, GCBcalc, stresscheck,
              and finalout
```

```
returns     : G array values
       Anew - cylinder area with buckled skin,
       Faxial, Faxialp - axial stress from F and Pa only, w/ sf and sfp
       gamF - general cylinder buckling knockdown factor for axial loads
       gamM - general cylinder buckling knockdown factor for bending
       GCB - flag for identifying the shell buckling failure mode
       i - index
       Io - cylinder total moment of inertia
       mdrv - the critical wavelength value for I-coupled buckling
       mgcb, ngcb - half axial and hoop waves for Nx
       N - knockdown adjusted applied line load
       Nx - critical general cylinder buckling line loads
       ncr - hoop waves for Pcr
       Pcr - critical general cylinder buckling pressure
       rasl, risl - radius average skin line, inner skin line
       St - applied tension stress
       Scol, Scrstcol - applied and critical stringer/column stress
       Scrip1/2, - crippling and local buckling critical stresses
       ScripS - weighted average of inelastic crippling stress
       Scrpl, Sskbd - skin stresses: crit. skin buckling, skin buck. driver
       Ssk - maximum stress in the skin
       tau, taucr, tauskbd, tau0 - shear stresses: maximum, skin buckling
              critical, skin buckling driver, minimum
       theta, dtheta - angle and angle between stringers
       Yst, Ysk - stringer and skin y distances from cylinder center
*************************************************************************/
void eval(struct stringer S, struct ring R, struct material M, struct load L,
       struct cylinder C, float G[], float *OBJ, FILE *out, int SSP,
       int mflag, char LEB);


/*************************************************************************/
/*************************    file CIO.C          ************************/
/*************************************************************************/

/* finalout prints the last set of output data
called from: eval
calls      :
returns    :
*************************************************************************/
void finalout(struct ring R, struct stringer S, struct material M,
       struct load L, struct cylinder C, float G[], float gamF, float gamM,
       char GCB, int mgcb, float N, int ncr, int ngcb, float Nx, float obj,
       FILE *out, float Pcrush, float Scol, float Scrip1, float Scrip2,
       float Scrstcol, char LEB, int mdrv, float Io, float Faxial);


/* cinput, finput, iinput read a character, a float and an integer,
       respectively from the input file then skips to the next line
called from: readinput
calls      :
returns    : a character, float, or integer
*************************************************************************/
char  cinput(FILE *infile);
float finput(FILE *infile);
int   iinput(FILE *infile);


/* pinput, repeats the information from the input file so that input and
       output are always kept together
called from: main
calls      :
returns    :
*************************************************************************/
void pinput(struct stringer S, struct ring R, struct material M,
       struct load L, struct cylinder C, long int IPRINT, long int METHOD,
       FILE *out, int SSP, float X[], float XL[], float XU[], int mflag,
```

```
        char LEB);


/* readinput, reads the information in the input file
called from: main
calls      :
returns    : all input information
*************************************************************************/
void readinput(char Title[], FILE *in, long int *IPRINT, long int *METHOD,
        int *SSP, struct material *M, struct cylinder *C, struct stringer *S,
        struct ring *R, struct load *L, float XL[], float X[], float XU[],
        float Tol[], int *mflag, char *LEB);


/*************************************************************************/
/************************    file CORSSTOL.C     ************************/
/*************************************************************************/

/* CALLevalCT arranges the CORSSTOL information and calls the analysis
called from: EVALTOL
calls      : eval
returns    : array of constraints for DOT
*************************************************************************/
void CALLevalCT(struct stringer S, struct ring R, struct material M,
        struct load L, struct cylinder C, float G[], float XW[],
        float *OBJ, FILE *out, int SSP, int mflag, char LEB);


/* corsstol controls the optimization loop of the sensitivity study
called from: main
calls      : CTPinput, EVALTOL, DOT, TOLOUT, and CTfinalout
returns    :
        INFO,IPRM,IWK,j,MINMAX,NRIWK,NRWK,RPRM,WK,OBJ - as in ctopt
        Gslope - d(constraint)/d(design variable)
        GW - array of worst condition constraints, used in DOT
        Wslope - d(weight)/d(design variable)
        Xnorm - array of design variable normalizing values
        AddTol - additonal tolerances gained by constraints not equal to 0
        delWt - delta weights from each design variables tolerance
        DVmin, DVmax - new design variable tolerance limits, based on AddTol
        OPTWT - the Minimum Material Condition weight
        NSTmin - the minimum number of stringers within the tolerance;
*************************************************************************/
void corsstol(struct stringer S, struct ring R, struct material M,
        struct load L, struct cylinder C, int hap, long int IPRINT,
        int l2ap, long int METHOD, long int NCON, long int NDV, int Nrng,
        int Nstap, FILE *out, int SSP, int tap, float Tol[], int tstap,
        int Wap, float X[], float XL[], float XU[],int mflag, char LEB);


/* CTPinput prints the tolerances to the output file
called from: corsstol
calls      :
returns    :
*************************************************************************/
void CTPinput(char stype, float Tol[], FILE *out);



/* CTfinalout prints the tolerancing information
called from: corsstol
calls      :
returns    :
        i,j - indices
        ERR - used to mark driving constraints
        string1,string2 - used for outputing diffent strings
        NSTmax - maximum number of stringers within the tolerance
```

```
*************************************************************************/
void CTfinalout(float OPTWT,    float MAXWT,
                float WSLOPE[], float DELWT[],     float Gslope[][6],
                float GW[],     float ADDTOL[][6], float DVMIN[],
                float DVMAX[],  char stype,        FILE *out,
                float Tol[],    float R,           float h,
                float 12,       float tst,         float b,
                float t,        float W, float Nst, float NSTmin,
                char LEB,       long int NCON);


/* EVALTOL controls the sensitivity analysis
called from: corsstol
calls      : CALLeval, WEIGHT
returns    : weight, GSLOPE, and GW
       f - worst case control flag
       G - working constraints
       Gsign - array indicating positive or negative slope of GSLOPE
       GWdone - completed GW calculating flags
       INITG - reference set of constraint values
       INITX - reference set of design varialbes
       j,i,k - indices
       OBJ - dummy weight variable
       XW - worst case set of design variables for a given constraint
*************************************************************************/
float EVALTOL(struct stringer S, struct ring R, struct material M,
       struct load L, struct cylinder C, float GSLOPE[][6], float GW[],
       int hap, int l2ap, int Nstap, FILE *out, int tap, float Tol[],
       int tstap, int Wap, float X[], float XNORM[], int SSP, int mflag,
       char LEB, long int NCON);


/* TOLOUT performs the sensitivity calculations
called from: corsstol
calls      : WEIGHT
returns    : AddTol, delWt, additional max and min design variables,
               the minimum material condition weight, and WSLOPE
       i,j - indices
       MAXWT - maximum material condition weight
       NEWDVS, NEWDVL - minimum and maximum holding variables for additional
              design variable range
       Nmin - used to calculate sensitivities to stringer spacing
       OBJH - comparison weight for calculating weight slopes
       TW - array of worst tolerances
       XW - working design variable array, Nst is replaced by b
*************************************************************************/
void TOLOUT(float ADDTOL[][6], float ALP,      float AR,  float DELWT[],
            float DVMAX[],     float DVMIN[],  float FWT, float L,
            float GSLOPE[][6], float GW[],     float L1,  float NRNG,
            float *OPTWT,       float R,       float RHO, char STYPE,
            float TOL[],       float WSLOPE[], float X[], float ZR,
            int harpos, int l2arpos, int tstarpos, int Nstarpos,
            int tarpos, int Warpos, float NSTmin, long int NCON);


/* WEIGHT calculates the weight of the cylinder
called from: EVALTOL, TOLOUT,
calls      :
returns    : weight of the structure
  float ALPHA,HA,AS,RASL,RISL;
       ALPHA - web angle in radians
       HA - web slant length
       AS - stringer cross sectional area
       RASL - radius at the center of the skin
       RISL - radius at the inner skin line
*************************************************************************/
float WEIGHT (float A,   float C,   float D,     float EE,    float FF,
```

```
            float FWT, float L,   float L1,    float NRNG,  float P,
            float R,   float RHO, float stype, float ALP,   float AR,
            float ZR);
```

File CTMAIN.C

```
#include "../corss/corss.h"

void main(argc, argv)
int argc;
char *argv[];
{
  FILE *in,*out;
  long int NDV, NCON, IPRINT, METHOD;
  float X[12],XL[6],XU[6],G[5],OBJ;
  float Xinit[12],Tol[6];
  char Title[80],LEB;
  int i,SSP,hap,l2ap,tstap,Nstap,tap,Wap,mflag;
  struct material M;
  struct load L;
  struct stringer S;
  struct ring R;
  struct cylinder C;
  printf("\nmain\n");
  NDV = 6;      NCON = 5;
  if ( (in = fopen(argv[1],"rt")) == NULL)
  {
    printf("\ncan't open input file, First parameter must be input file");
    comlineerr();
  }
  if (strcmp(argv[1],argv[2]) == 0)
  {
    printf("\ninput filename is the same as output filename");
    comlineerr();
  }
  if ( (out = fopen(argv[2],"wt")) == NULL)
  {
    printf("\ncan't open output file, Second param. must be output file");
    comlineerr();
  }
  fprintf(out,"CORSS - Cylinder Optimization of Rings, Skin and Stringe");
  fprintf(out,"rs\n     NASA/MSFC/ED52 - Structural Development Branch");
  fprintf(out,"\n     Jeff Finckenor, Sep. 1993, ver. 2.1");
  fprintf(out,"\n     Copyright (c) 1994 National Aeronautics and");
  fprintf(out,"\n     Space Administration. No copyright claimed in");
  fprintf(out,"\n     USA under Title 17, U.S. Code. All other rights");
  fprintf(out,"\n     reserved.\n");
  printf("CORSS - Cylinder Optimization of Rings, Skin and Stringe");
  printf("rs\n     NASA/MSFC/ED52 - Structural Development Branch");
  printf("\n     Jeff Finckenor, Sep. 1993, ver. 2.1");
  printf("\n     Copyright (c) 1994 National Aeronautics and");
  printf("\n     Space Administration. No copyright claimed in");
  printf("\n     USA under Title 17, U.S. Code. All other rights");
  printf("\n     reserved.\n\n");
  readinput(Title,in,&IPRINT,&METHOD,&SSP,&M,&C,&S,&R,&L,XL,X,XU,Tol,&mflag,&LEB);
  fclose(in);
  fprintf(out,"\n%s",Title);
  if (0 == R.N) { R.A = .001;  R.I = .001;  R.Z = .001;  R.J = .001; }
  pinput(S,R,M,L,C,IPRINT,METHOD,out,SSP,X,XL,XU,mflag,LEB);
  if ( (S.stype != 'H') && (S.stype != 'I') )
  {
    printf("\n\n%c is not a valid stringer type\a\a\a",S.stype);
```

```
    exit(4);
  }
  if ( ('H'==S.stype) && ('N'!=LEB) ) NCON--;
  NDV = initDVs(XL,X,XU,&hap,&12ap,&tstap,&Nstap,&tap,&Wap,Xinit,NDV);
  if (NDV != 0)
       ctopt(SSP,out,X,Xinit,hap,12ap,tstap,Nstap,tap,Wap,S,R,M,L,C,G,
                &OBJ,&METHOD,&IPRINT,&NDV,&NCON,XL,XU,mflag,LEB);
  fprintf(out,"\n");
  fputs(Title,out);
  if ( (SSP == 1) || (SSP == 3) ) SSP = 4; else SSP = 5;
  CALLeval(S,R,M,L,C,G,X,Xinit,&OBJ,out,SSP,hap,12ap,Nstap,tap,
                                tstap,Wap,mflag,LEB);
  for (i=0; i<NCON; i++)
    if (G[i]>0)
    {
      fprintf(out,"\n\n** WARNING G[%d] (=%7.5f>0) IS NOT ",i,G[i]);
      fprintf(out,"SATISFIED, THIS IS NOT A VALID SOLUTION!");
    }
  if ( (Tol[0]!=0) || (Tol[1]!=0) || (Tol[2]!=0) || (Tol[3]!=0) ||
       (Tol[4]!=0) || (Tol[5]!=0) )
  {
    for (i=0; i<NDV; i++)
      {  XL[i] *= Xinit[i];  XU[i] *= Xinit[i];  X[i]  *= Xinit[i];  }
    corsstol(S,R,M,L,C,hap,IPRINT,12ap,METHOD,NCON,NDV,R.N,Nstap,out,
       SSP,tap,Tol,tstap,Wap,X,XL,XU,mflag,LEB);
       /* corsstol must define IPRM, IWK, MINMAX, NRIWK, NRWK, RPRM, WK */
  }
  fclose(out);
  printf("\a\a\a");
}  /*************************************************************  end main  */
```

File CT.C

```
#include "..\corss\corss.h"

void corsstol(struct stringer S, struct ring R, struct material M,
       struct load L, struct cylinder C, int hap, long int IPRINT,
       int 12ap, long int METHOD, long int NCON, long int NDV, int Nrng,
       int Nstap, FILE *out, int SSP, int tap, float Tol[], int tstap,
       int Wap, float X[], float XL[], float XU[],int mflag, char LEB)
{
  long int INFO,IPRM[20],IWK[110],j,MINMAX,NRIWK,NRWK;
  float RPRM[20],WK[450];
  float Gslope[5][6], GW[5], Wslope[6], Xnorm[12], AddTol[5][6], delWt[6],
       DVmin[6], DVmax[6], OBJ, OPTWT, NSTmin;
  int i;
  SSP = 0;
  for (j=0; j<20; j++)
  {
   RPRM[j] = 0.0;
   IPRM[j] = 0;
  }
  NRWK =   450;         /*  make the dimension of IWK and WK the same  */
  NRIWK =  110;
  MINMAX = -1;
  INFO = 0;
  CTPinput(S.stype,Tol,out);
  for (i=0; i<NDV; i++)
  {
    Xnorm[i] = X[i];
    XL[i] = XL[i]/Xnorm[i];
    XU[i] = XU[i]/Xnorm[i];
    X[i] = 1.;
  }
  for (i=NDV; i<12; i++) Xnorm[i]=1.;
```

```
  if (NDV==0)
    OBJ = EVALTOL(S,R,M,L,C,Gslope,GW,hap,12ap,Nstap,out,tap,Tol,tstap,
              Wap,X,Xnorm,SSP,mflag,LEB,NCON);
  else
  {
    INFO = 0;
    do
    {
      OBJ = EVALTOL(S,R,M,L,C,Gslope,GW,hap,12ap,Nstap,out,tap,Tol,tstap,Wap,
              X,Xnorm,SSP,mflag,LEB,NCON);
      DOT(&INFO,&METHOD,&IPRINT,&NDV,&NCON,X,XL,XU,&OBJ,&MINMAX,GW,RPRM,IPRM,
              WK,&NRWK,IWK,&NRIWK);
    } while (INFO!=0);
  }
  printf("\nToleranced Optimization Complete, Max Weight = %f",OBJ);
  for (i=0; i<NDV; i++) X[i] *= Xnorm[i];
  NSTmin = 2*pi*C.r/(2*pi*C.r/X[Nstap]+Tol[3]);
  for (i=0; i<NCON; i++) GW[i] -= 1.E-5;
  TOLOUT(AddTol,S.alp,R.A,delWt,DVmax,DVmin,C.fwt,C.l,Gslope,GW,S.ll,
       R.N,&OPTWT,C.r,M.rho,S.stype,Tol,Wslope,X,R.Z,hap,12ap,tstap,
       Nstap,tap,Wap,NSTmin,NCON);
  CTfinalout(OPTWT,OBJ,Wslope,delWt,Gslope,GW,AddTol,DVmin,DVmax,S.stype,
       out,Tol,C.r,X[hap],X[12ap],X[tstap],2*pi*C.r/X[Nstap],
       X[tap],X[Wap],X[Nstap],NSTmin,LEB,NCON);
}       /****************** end CORSSTOL ****************/


float WEIGHT (float A,   float C,   float D,     float EE,    float FF,
              float FWT, float L,   float L1,    float NRNG,  float P,
              float R,   float RHO, float stype, float ALP,   float AR,
              float ZR)
{
  float ALPHA,HA,AS,RASL,RISL;
  ALPHA = (ALP*pi)/180.0;
  if ('H'==stype)
  {
    HA = (A-D-P)/(cos(ALPHA));
    AS = (2.0*(L1*D)) + (2.0*(HA*D)) + (C*P);
  }
  else
  {
    HA = (A-C-L1)/(cos(ALPHA));
    AS = (P*L1) + (D*HA) + (P*C);
  }
  RASL = R-(FF/2.0);
  RISL = R-FF;
  return (((pi*2.0*RASL*FF*L) + (AS*L*EE) + ((AR*NRNG*2.0*pi) *
              (RISL+ZR)))*RHO + FWT);
}       /************** end weight ********************/


float EVALTOL(struct stringer S, struct ring R, struct material M,
       struct load L, struct cylinder C, float GSLOPE[][6], float GW[],
       int hap, int 12ap, int Nstap, FILE *out, int tap, float Tol[],
       int tstap, int Wap, float X[], float XNORM[], int SSP, int mflag,
       char LEB, long int NCON)
{
  float INITG[5],G[5],INITX[6],XW[6],OBJ,Gsign[5];
  int j,i,k,f,GWdone[5];
  printf("\nEVALTOL");
  XW[0] = INITX[0] = X[hap]*XNORM[hap];
  XW[1] = INITX[1] = X[12ap]*XNORM[12ap];
  XW[2] = INITX[2] = X[tstap]*XNORM[tstap];
  XW[3] = INITX[3] = 2*pi*C.r/(X[Nstap]*XNORM[Nstap]);
  XW[4] = INITX[4] = X[tap]*XNORM[tap];
```

```
 XW[5] = INITX[5] = X[Wap]*XNORM[Wap];
 for (i=0; i<NCON; i++) Gsign[i]=0;
 CALLevalCT(S,R,M,L,C,INITG,INITX,&OBJ,out,SSP,mflag,LEB);
 printf(" GSLOPE");
 for (j=0; j<6; j++)
 {
   XW[j] *= 1.01;
   CALLevalCT(S,R,M,L,C,G,XW,&OBJ,out,SSP,mflag,LEB);
   for (i=0; i<NCON; i++)
   {
     GSLOPE[i][j] = (G[i]-INITG[i])/(XW[j]-INITX[j]);
     if (GSLOPE[i][j]>0) Gsign[i]+=pow(10,j);
   }
   XW[j] /= 1.01;
 }
 printf(" GWorst");
 for (k=0; k<NCON; k++) GWdone[k] = 0;
 for (i=0; i<NCON; i++)
 {
   f = 0;
   for (k=0; k<i; k++) if (Gsign[k]==Gsign[i]) f++;
   if (0==f)
   {
     for (j=0; j<6; j++)
      if (GSLOPE[i][j]<=0) XW[j] = INITX[j];
                     else XW[j] = INITX[j]+Tol[j];
     CALLevalCT(S,R,M,L,C,G,XW,&OBJ,out,SSP,mflag,LEB);
     GW[i] = G[i] + 1.E-5;
     GWdone[i] = 1;
     for (k=i+1; k<NCON; k++)
      if ( (Gsign[k]==Gsign[i]) && (GWdone[k]==0) )
      {
        GW[k] = G[k]+.005;
        GWdone[k] = 1;
      }
   }   /** end if f==0 **/
 }             /* end for i */
 return ( WEIGHT(INITX[0]+Tol[0],INITX[1]+Tol[1],INITX[2]+Tol[2],
             2*pi*C.r/INITX[3],INITX[4]+Tol[4],C.fwt,C.l,S.l1,R.N,
             INITX[5]+Tol[5],C.r,M.rho,S.stype,S.alp,R.A,R.Z) );
}       /******************** end evaltol ************************/


void CALLevalCT(struct stringer S, struct ring R, struct material M,
       struct load L, struct cylinder C, float G[], float XW[],
       float *OBJ, FILE *out, int SSP, int mflag, char LEB)
{
  S.h = XW[0];
  S.l2 = XW[1];
  S.N = 2*pi*C.r/XW[3];
  S.t = XW[2];
  C.t = XW[4];
  S.w = XW[5];
  eval(S,R,M,L,C,G,OBJ,out,SSP,mflag,LEB);
}       /****************************************** end CALLevalCT ******/


void TOLOUT(float ADDTOL[][6], float ALP,      float AR,  float DELWT[],
            float DVMAX[],     float DVMIN[],  float FWT, float L,
            float GSLOPE[][6], float GW[],     float L1,  float NRNG,
            float *OPTWT,       float R,        float RHO, char STYPE,
            float TOL[],       float WSLOPE[], float X[], float ZR,
            int harpos, int l2arpos, int tstarpos, int Nstarpos,
            int tarpos, int Warpos, float NSTmin, long int NCON)
{
```

```
float TW[6], OBJH, NEWDVS, NEWDVL, XW[6], MAXWT, Nmin;
int i,j;
printf("\nTOLOUT");
XW[0] = X[harpos];
XW[1] = X[12arpos];
XW[2] = X[tstarpos];
XW[3] = 2*pi*R/X[Nstarpos];
XW[4] = X[tarpos];
XW[5] = X[Warpos];
*OPTWT = WEIGHT(XW[0],XW[1],XW[2],NSTmin,XW[4],FWT,L,L1,NRNG,XW[5],
              R,RHO,STYPE,ALP,AR,ZR);
printf(" WSLOPE and DELWT");
printf("\nWSLOPE about maximum (for comparison only)");
for (i=0; i<6; i++) TW[i] = TOL[i];
MAXWT = WEIGHT(XW[0]+TOL[0],XW[1]+TOL[1],XW[2]+TOL[2],X[Nstarpos],
          XW[4]+TOL[4],FWT,L,L1,NRNG,XW[5]+TOL[5],R,RHO,STYPE,ALP,AR,ZR);
for (i=0; i<6; i++)
{
  if (3==i)
  {
    if (TOL[i]!=0)
    {
     OBJH = WEIGHT(XW[0]+TOL[0],XW[1]+TOL[1],XW[2]+TOL[2],NSTmin,
           XW[4]+TOL[4],FWT,L,L1,NRNG,XW[5]+TOL[5],R,RHO,STYPE,ALP,AR,ZR);
     DELWT[i] = MAXWT - OBJH;
     WSLOPE[i] = DELWT[i]/(X[Nstarpos]-NSTmin);
    }
    else
    {
     OBJH = WEIGHT(XW[0]+TOL[0],XW[1]+TOL[1],XW[2]+TOL[2],.99*X[Nstarpos],
           XW[4]+TOL[4],FWT,L,L1,NRNG,XW[5]+TOL[5],R,RHO,STYPE,ALP,AR,ZR);
     DELWT[i] = MAXWT - OBJH;
     WSLOPE[i] = DELWT[i]/(.01*X[Nstarpos]);
    }
  }
  else
  {
    if (TOL[i]==0)  TW[i] = -.01*XW[i]; else TW[i] = 0;
    OBJH = WEIGHT(XW[0]+TW[0],XW[1]+TW[1],XW[2]+TW[2],X[Nstarpos],
           XW[4]+TW[4],FWT,L,L1,NRNG,XW[5]+TW[5],R,RHO,STYPE,ALP,AR,ZR);
    DELWT[i] = MAXWT - OBJH;
    if (TOL[i]==0) WSLOPE[i] = DELWT[i]/(.01*XW[i]);
     else WSLOPE[i] = DELWT[i]/TOL[i];
  }
  if (TOL[i]==0) DELWT[i]=0;
  printf("\nDV=i=%d, WTi=%8.2f, DELWTi=%8.2f, TOLi=%7.5f, WSLOPEi=%9.2f",
           i,OBJH,DELWT[i],TOL[i],WSLOPE[i]);
  TW[i] = TOL[i];
}     /* end for i */
printf("\nWSLOPE about optimum (used in calculations)");
for (i=0; i<6; i++) TW[i] = 0;
for (i=0; i<6; i++)
{
  if (i==3)
  {
    if (TOL[i]!=0)
    {
     OBJH = WEIGHT(XW[0],XW[1],XW[2],X[Nstarpos],
           XW[4],FWT,L,L1,NRNG,XW[5],R,RHO,STYPE,ALP,AR,ZR);
     TW[i] = X[Nstarpos]-NSTmin;
    }
    else
    {
     OBJH = WEIGHT(XW[0],XW[1],XW[2],1.01*X[Nstarpos],
           XW[4],FWT,L,L1,NRNG,XW[5],R,RHO,STYPE,ALP,AR,ZR);
     TW[i] = .01*X[Nstarpos];
    }
```

```
    }
    else
    {
      if (TOL[i]!=0) TW[i] = TOL[i]; else TW[i] = .01*XW[i];
      OBJH = WEIGHT(XW[0]+TW[0],XW[1]+TW[1],XW[2]+TW[2],NSTmin,
             XW[4]+TW[4],FWT,L,L1,NRNG,XW[5]+TW[5],R,RHO,STYPE,ALP,AR,ZR);
    }
    DELWT[i] = OBJH - *OPTWT;
    WSLOPE[i] = DELWT[i]/TW[i];
    if (TOL[i]==0) { DELWT[i]=0;  TW[i]=0; }
    TW[3]=TOL[3];
    printf("\nDV=i=%d, WTi=%8.2f, DELWTi=%8.2f, TOLi=%7.5f, WSLOPEi=%9.2f",
              i,OBJH,DELWT[i],TW[i], WSLOPE[i]);
    TW[i] = 0;
  }     /* end for i */
  printf("\nCalculate ADDTOL");
  for (j=0; j<6; j++)
  {
    for (i=0; i<NCON; i++)
    {
      if (fabs(GSLOPE[i][j])<1e-5) ADDTOL[i][j] = -99;
       else ADDTOL[i][j] = XW[j] - GW[i]/GSLOPE[i][j];
      if (GSLOPE[i][j]>0) ADDTOL[i][j] += TOL[j];
    }   /* end for i */
  }     /* end for j */
  printf(" DVMIN and DVMAX");
  for (j=0; j<6; j++)
  {
    NEWDVS =    0.0;
    NEWDVL = 9999.0;
    for (i=0; i<NCON; i++)
    {
      if ( (ADDTOL[i][j]<XW[j]) && (ADDTOL[i][j]>NEWDVS) )
       NEWDVS = ADDTOL[i][j];
      if ( (ADDTOL[i][j]>(XW[j]+TOL[j])) && (ADDTOL[i][j]<NEWDVL) )
       NEWDVL = ADDTOL[i][j];
    }   /* end for i */
    DVMIN[j] = NEWDVS;
    DVMAX[j] = NEWDVL;
  }     /* end for j */
}       /************************* end tolout ********/


void CTPinput(char stype, float Tol[], FILE *out)
{
  fprintf(out,"\n\n\n---Tolerance Sensitivity Output for CORSS---");
  fprintf(out,"\n\nDesign Variable Tolerances");
  fprintf(out,"\n     htol    = %20.4f Stringer Height Tolerance",Tol[0]);
  if ('H'==stype)
  {
    fprintf(out,"\n     12tol   = %20.4f Top Flange Length Tolerance",Tol[1]);
    fprintf(out,"\n     tsttol  = %20.4f Stringer Thickness Tolerance",Tol[2]);
  }
  else
  {
    fprintf(out,"\n     12tol   = %20.4f Top Flange Thickness Tolerance",Tol[1]);
    fprintf(out,"\n     tsttol  = %20.4f Stringer Web Thickness Tolerance",Tol[2]);
  }
  fprintf(out,"\n     Nsttol  = %20.4f Stringer Spacing Tolerance",Tol[3]);
  fprintf(out,"\n     ttol    = %20.4f Skin Thickness Tolerance",Tol[4]);
  if ('H'==stype)
    fprintf(out,"\n     Wtol    = %20.4f Top Flange Thickness Tolerance",Tol[5]);
  else
    fprintf(out,"\n     Wtol    = %20.4f Top Flange Width Tolerance",Tol[5]);
  fprintf(out,"\n\n\nNEW OPTIMUM ANALYSIS");
}       /******************* end CTpinput ********************/
```

```
void CTfinalout(float OPTWT,    float MAXWT,
                float WSLOPE[], float DELWT[],     float Gslope[][6],
                float GW[],     float ADDTOL[][6], float DVMIN[],
                float DVMAX[],  char stype,        FILE *out,
                float Tol[],    float R,           float h,
                float l2,       float tst,         float b,
                float t,        float W, float Nst, float NSTmin,
                char LEB,       long int NCON)
{
  int i,j;
  char ERR[5], string1[80], string2[80];
  float NSTmax;
  fprintf(out,"\n\n\nSENSITIVITY ANALYSIS RESULTS");
  fprintf(out,"\n\n\t\t\tOptimum    Maximum      Delta");
  fprintf(out,"\nCylinder Weight\t%15.2f %10.2f %10.2f",OPTWT,MAXWT,MAXWT-OPTWT);
  fprintf(out,"\n\nDesign Variable Solutions");
  fprintf(out,"\n     DV        Optimum       OPT+TOL       dWt/dDV           dWt");
  fprintf(out,"\n     h   =  %10.5f    %10.5f    %10.4f    %10.4f",
       h,h+Tol[0],WSLOPE[0],DELWT[0]);
  fprintf(out,"\n     l2  =  %10.5f    %10.5f    %10.4f    %10.4f",
       l2,l2+Tol[1],WSLOPE[1],DELWT[1]);
  fprintf(out,"\n     tst =  %10.5f    %10.5f    %10.4f    %10.4f",
       tst,tst+Tol[2],WSLOPE[2],DELWT[2]);
  fprintf(out,"\n    (b   =  %10.5f    %10.5f)",b+Tol[3],b);
  fprintf(out,"\n     Nst =  %10.5f    %10.5f    %10.4f    %10.4f",
       NSTmin,Nst,WSLOPE[3],DELWT[3]);
  fprintf(out,"\n     t   =  %10.5f    %10.5f    %10.4f    %10.4f",
       t,t+Tol[4],WSLOPE[4],DELWT[4]);
  fprintf(out,"\n     W   =  %10.5f    %10.5f    %10.4f    %10.4f",
       W,W+Tol[5],WSLOPE[5],DELWT[5]);
  fprintf(out,"\n\nNumber of Stringers, Nst = 2*pi*r/b, b = stringer spacing");
  fprintf(out,"\n\nSmall changes in b change the theoretical number of ");
  fprintf(out,"stringers used in\nthe analysis. As stringer spacing ");
  fprintf(out,"increases, the number of stringers\ndecreases. The design ");
  fprintf(out,"will have an integer number of stringers, but\nthe effect ");
  fprintf(out,"of the spacing tolerance is shown here.");
  fprintf(out,"  Any differences\nbetween the sum of the dWt values ");
  fprintf(out,"and deltaWt are due to the weight\nfunction not being linear.");
  fprintf(out,"  All calculations are based on taking\ndifferences about ");
  fprintf(out,"the optimum.");
  fprintf(out,"\n\nGSLOPE ARRAY");
  fprintf(out,"\nDV   dSkB/dDV  dShB/dDV  dStr/dDV  ");
  if (stype=='H')
    if (LEB=='Y') fprintf(out,"dCrp/dDV");
      else fprintf(out,"dLBF/dDV  dLBW/dDV");
  else
    if (LEB=='Y') fprintf(out,"dCrp/dDV  dICB/dDV");
      else fprintf(out,"dLBF/dDV  dICB/dDV");
  fprintf(out,"\nh  ");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][0]);
  fprintf(out,"\nl2 ");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][1]);
  fprintf(out,"\ntst");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][2]);
  fprintf(out,"\nb  ");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][3]);
  fprintf(out,"\nt  ");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][4]);
  fprintf(out,"\nW  ");
  for (i=0; i<NCON; i++) fprintf(out," %9.4f",Gslope[i][5]);
  /* determine critical constraints */
  for (i=0; i<NCON; i++) if (CW[i]>-.05) ERR[i]='*'; else ERR[i]=' ';
  fprintf(out,"\n\nG[i]");
  for (i=0; i<NCON; i++) fprirtf(out," %8.5f%c",GW[i],ERR[i]);
```

```
fprintf(out,"\n\n* Denotes the Driving Constraints, which must be negative to ");
fprintf(out,"be satisfied\nGSLOPE = [G(1.01*DV)-G(DV)]/(1.01*DV-DV)");
fprintf(out,"\nnegative values of d( )/dDV indicate a safer design for an ");
fprintf(out,"increase in DV\nDV = Design Variable\t\tG[i] = Constraint Value");
fprintf(out,"\nSkB = Skin Buckling\t\tShB = Shell Buckling");
fprintf(out,"\nStr = Stress");
if (stype=='H')
  if (LEB=='Y') fprintf(out,"\t\t\tCrp = Stringer Crippling");
    else fprintf(out,"\nLBF = Local Buckling, Flange\tLBW = Local Buckling, Web");
else
  if (LEB=='Y') fprintf(out,
        "\t\t\tCrp = Stringer Crippling\nICB = 'I'-stringer Coupled Buckling");
    else fprintf(out,
        "\nLBF = Local Buckling, Flannge\tICB = 'I'-stringer Coupled Buckling");
fprintf(out,"\n\nADDITIONAL TOLERANCE ARRAY");
fprintf(out,"\nDV        SkB       ShB       Str       ");
if (stype=='H')
  if (LEB=='Y') fprintf(out,"Crp       ");
    else fprintf(out,"LBF       LBW");
else
  if (LEB=='Y') fprintf(out,"Crp       ICB");
    else fprintf(out,"LBF       ICB");
fprintf(out,"\nh  ");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][0]);
fprintf(out,"\n12 ");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][1]);
fprintf(out,"\ntst");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][2]);
fprintf(out,"\nb  ");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][3]);
fprintf(out,"\nt  ");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][4]);
fprintf(out,"\nW  ");
for (i=0; i<NCON; i++) fprintf(out," %9.4f",ADDTOL[i][5]);
fprintf(out,"\n\nThe ADDTOL array contains the values of the design ");
fprintf(out,"variables\nat which the constraint is equal to 0,");
fprintf(out," assuming linear relationships.");
fprintf(out,"\nADDTOL = DV - G(DV)/GSLOPE");
fprintf(out,"\nNegative values should be disregarded, since they ");
fprintf(out,"indicate the design\nvariable can be brought past zero ");
fprintf(out,"without violating that particular\nconstraint.");
fprintf(out,"\n\n\nNEW RANGES for the Design Variables");
fprintf(out,"     (from the Additonal Tolerance array)");
fprintf(out,"\n\n               Below Optimum ----               Above ");
fprintf(out,"Maximum ----\n       Lower   Weight     Added       Upper    ");
fprintf(out," Weight      Added     New Tol.\nDV      Limit   Savings    ");
fprintf(out," Toler.     Limit    Penalty     Toler.   Band");
sprintf(string2,"    No constraint limits the upper bound");
fprintf(out,"\nh   %9.5f %9.4f %9.5f ",
                  DVMIN[0],(h-DVMIN[0])*WSLOPE[0],h-DVMIN[0]);
if (DVMAX[0]==9999) fprintf(out,"%s",string2);
  else fprintf(out,"%10.5f %10.4G %10.5f %10.5f",DVMAX[0],
       (DVMAX[0]-Tol[0]-h)*WSLOPE[0],DVMAX[0]-Tol[0]-h,DVMAX[0]-DVMIN[0]);
fprintf(out,"\n12  %9.5f %9.4f %9.5f ",
                  DVMIN[1],(12-DVMIN[1])*WSLOPE[1],12-DVMIN[1]);
if (DVMAX[1]==9999) fprintf(out,"%s",string2);
  else fprintf(out,"%10.5f %10.4G %10.5f %10.5f",DVMAX[1],
     (DVMAX[1]-Tol[1]-12)*WSLOPE[1],DVMAX[1]-Tol[1]-12,DVMAX[1]-DVMIN[1]);
fprintf(out,"\ntst %9.5f %9.4f %9.5f ",
             DVMIN[2],(tst-DVMIN[2])*WSLOPE[2],tst-DVMIN[2]);
if (DVMAX[2]==9999) fprintf(out,"%s",string2);
  else fprintf(out,"%10.5f %10.4G %10.5f %10.5f",DVMAX[2],
    (DVMAX[2]-Tol[2]-tst)*WSLOPE[2],DVMAX[2]-Tol[2]-tst,DVMAX[2]-DVMIN[2]);
if (DVMIN[3]<1e-4) NSTmax = 9999; else NSTmax = 2*pi*R/DVMIN[3];
fprintf(out,"\nNst %9.5f %9.4f          ",
               2*pi*R/DVMAX[3],(NSTmin-2*pi*R/DVMAX[3])*WSLOPE[3]);
if (NSTmax==9999) fprintf(out,"%s",string2);
```

```
    else fprintf(out," %10.5f %10.4G",NSTmax,(NSTmax-Nst)*WSLOPE[3]);
  if (DVMIN[3]==0) fprintf(out,"\n(b    No limit on the lower bound");
    else
      fprintf(out,
          "\n(b  %9.5f           %9.5f ",DVMIN[3], b-DVMIN[3]);
  fprintf(out," %10.5f            %10.5f %10.5f)",
       DVMAX[3], DVMAX[3]-Tol[3]-b,DVMAX[3]-DVMIN[3]);
  fprintf(out,"\nt   %9.5f %9.4f %9.5f ",
      DVMIN[4],(t-DVMIN[4])*WSLOPE[4],t-DVMIN[4]);
  if (DVMAX[4]==9999) fprintf(out,"%s",string2);
    else fprintf(out,"%10.5f %10.4G %10.5f %10.5f",DVMAX[4],
       (DVMAX[4]-Tol[4]-t)*WSLOPE[4],DVMAX[4]-Tol[4]-t,DVMAX[4]-DVMIN[4]);
  fprintf(out,"\nW   %9.5f %9.4f %9.5f ",
      DVMIN[5],(W-DVMIN[5])*WSLOPE[5],W-DVMIN[5]);
  if (DVMAX[5]==9999) fprintf(out,"%s",string2);
    else fprintf(out,"%10.5f %10.4G %10.5f %10.5f",DVMAX[5],
       (DVMAX[5]-Tol[5]-W)*WSLOPE[5],DVMAX[5]-Tol[5]-W,DVMAX[5]-DVMIN[5]);
  fprintf(out,"\n\nThe only penalty for increasing design variables with");
  fprintf(out," no upper limit is an\nincrease in weight. Note that an ");
  fprintf(out,"unsatisfied constraint (G[i]>0) will\ncause unrealistic ");
  fprintf(out,"results. For a feasible solution, no constraints will be");
  fprintf(out,"\nviolated as long as the DVs are held within the limits ");
  fprintf(out,"specified in this\nchart when");
  fprintf(out," ASSUMING LINEAR RELATIONSHIPS.");
}       /*************** end CTfinalout *****************/
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collecton of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, Va 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1995	Technical Paper

4. TITLE AND SUBTITLE

CORSSTOL: Cylinder Optimization of Rings, Skin, and Stringers With Tolerance Sensitivity

5. FUNDING NUMBERS

6. AUTHOR(S)

J. Finckenor and M. Bevill

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

8. PERFORMING ORGANIZATON REPORT NUMBERS

M-779

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TP-3551

11. SUPPLEMENTARY NOTES

Prepared by Structures and Dynamics Laboratory, Science and Engineering Directorate.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified

Subject category 64

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Cylinder Optimization of Rings, Skin, and Stringers with Tolerance (CORSSTOL) sensitivity is a design optimization program incorporating a method to examine the effects of user-provided manufacturing tolerances on weight and failure. CORSSTOL gives designers a tool to determine tolerances based on need. This is a decisive way to choose the best design among several manufacturing methods with differing capabilities and costs.

CORSSTOL initially optimizes a stringer-stiffened cylinder for weight without tolerances. The skin and stringer geometry are varied, subject to stress and buckling constraints. Then the same analysis and optimization routines are used to minimize the maximum material condition weight subject to the least favorable combination of tolerances. The adjusted optimum dimensions are provided with the weight and constraint sensitivities of each design variable. The designer can immediately identify critical tolerances. The safety of parts made out of tolerance can also be determined.

During design and development of weight-critical systems, design/analysis tools that provide product-oriented results are of vital significance. The development of this program and methodology provides designers with an effective cost- and weight-saving design tool. The tolerance sensitivity method can be applied to any system defined by a set of deterministic equations.

14. SUBJECT TERMS

optimization, tolerancing, manufacturing

15. NUMBER OF PAGES

36

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclasssified	Unlimited